Functions, Objects, Events

Functions, Arrow Functions, Callbacks, Objects, Classes, Constructors, Methods, Events

Svetlin Nakov, PhD

Co-founder @ SoftUni

SoftUni AI https://ai.softuni.bg

Agenda

SoftUni AI

1. **Functions** in JS
   - Functions, Parameters, Return Value, Defining and Invoking
   - Arrow Functions, Functions as Parameters, Callbacks
2. **Objects** in JS
   - Keeping Related Values Together
3. Intro to **Classes** in JS: Defining and Using Classes
   - Classes, Fields, Constructors, Methods
4. **Events** and **Event Handling**
   - Event Handling in HTML and JavaScript

Sli.do Code

#AI-Programming

Join at

slido.com

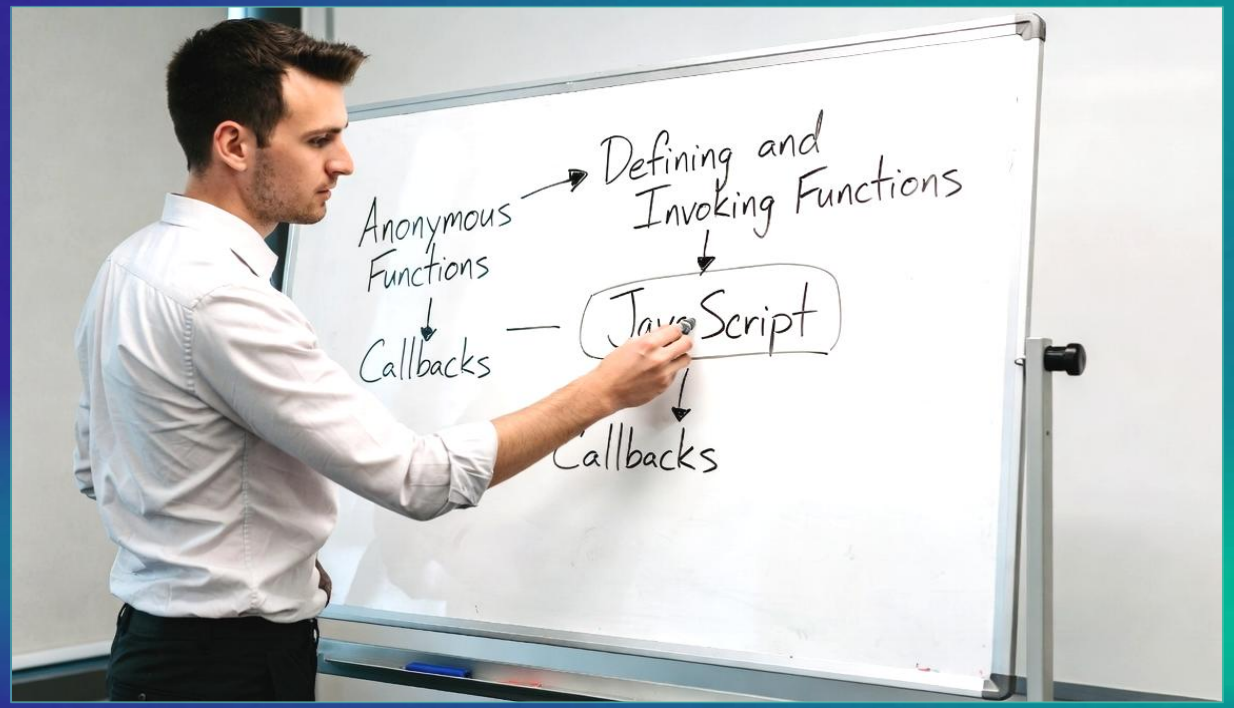
#AI-Programming

Breaks

20:00 / 21:00

Functions in JS

Defining and Invoking Functions, Recursive Functions, Debugging, Variable Arguments



Functions in JavaScript

SoftUni AI

- In JavaScript, a **function** is **named block of code**

```
function printHello() {
  console.log("Hello!");
  console.log("I am a function.");
}
```

- Once defined, a function can be **invoked** multiple times

```
for (let i = 1; i < 10; i++) {
  printHello();
}
```

Functions in JavaScript

SoftUni AI

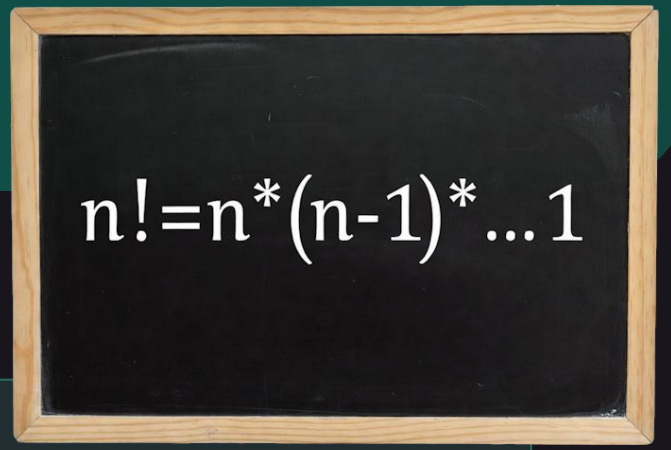
- Functions can take **parameters** and **return** values

```
function calcCircleArea(radius) {
  let area = Math.PI * radius * radius;
  return area;
}
```

- Once defined, a function can be **invoked** multiple times with different parameters

```
console.log(calcCircleArea(5));
console.log(calcCircleArea(12.5));
```

Recursive Functions



- A function can **invoke itself** recursively:

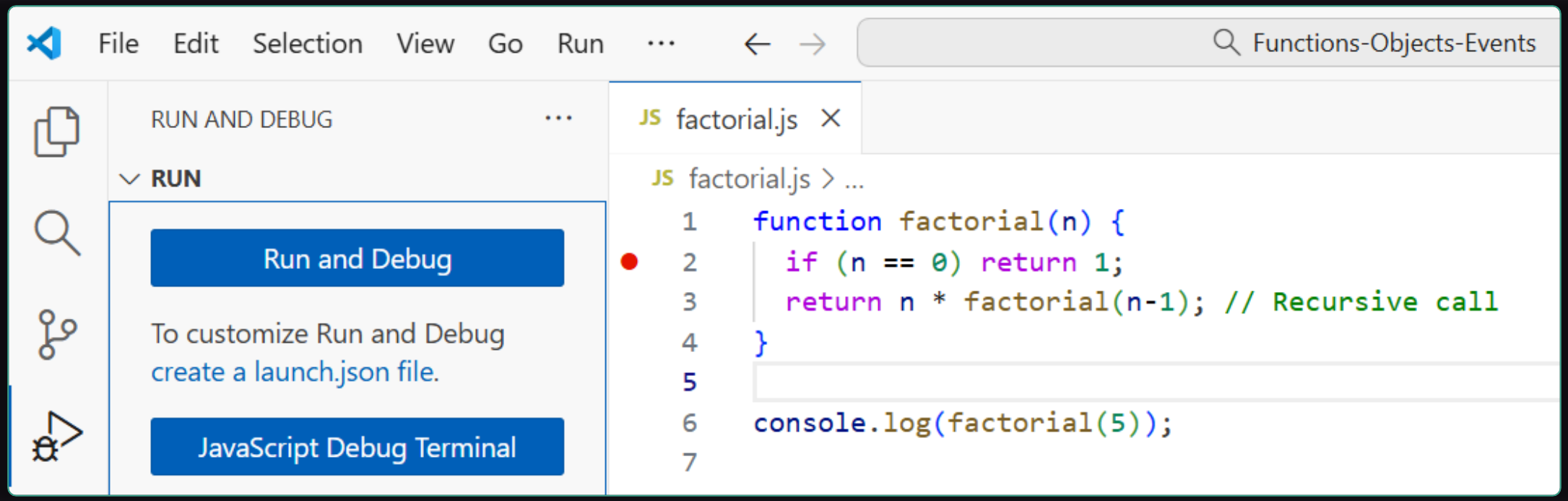
```
function factorial(n) {
  if (n == 0) return 1; // Base case
  return n * factorial(n-1); // Recursive call
}

console.log(factorial(5)); // 120
```

- **Recursion** is a powerful technique in programming
- Recursive functions should have a **base case** to avoid **infinite recursion**

Debugging & Breakpoints in VS Code

SoftUni AI

- **Debugging** == tracing the code execution to find bugs

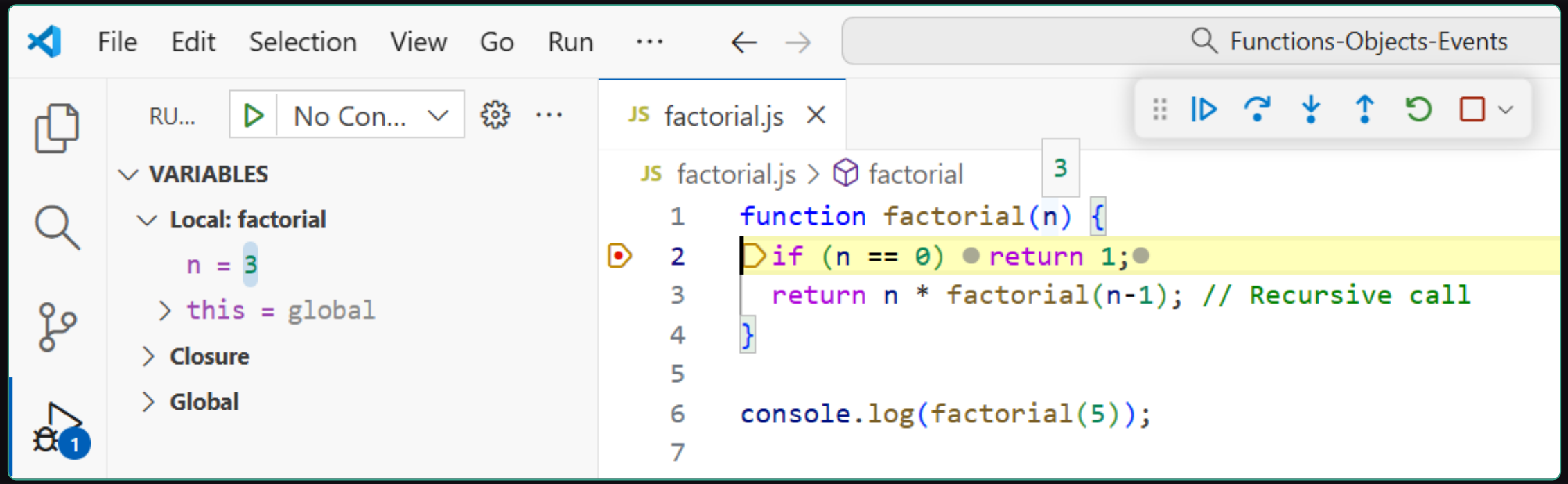


- **Breakpoint** == intentional **pause-point** in the code
  - Stops the code execution to inspect the internal state

Using the VS Code Debugger

SoftUni AI

- Inspecting the internal **execution state**



- **Watch** / modify variables, view the **call stack**
- **[F5]** → Continue, **[F10]** → Step Over, **[F11]** → Step Into

Problem: Big Factorial

- Write a JS function for **calculate n!** (factorial)
  - Ensure it works for **big inputs** (e. g. 50 factorial)
- **Solution**: we shall use **BigInt** arithmetic

```
function factorial(n) {
  if (n == 0 || n == 1) return 1n;
  return BigInt(n) * factorial(n - 1);
}
console.log(String(factorial(50)));
// 30414093201713378043612608166064768844377641568960512000000000000
```

n!

Judge link: https://alpha.judge.softuni.org/contests/functions-objects-events/5273

Variable Number of Arguments

SoftUni AI

```
function sum(...numbers) {
  let total = 0;
  for (let num of numbers)
    total += num;
  return total;
}

console.log(sum(2, 3)); // 5
console.log(sum(2, 3, 4, 5)); // 14
console.log(sum(3)); // 3
console.log(sum()); // 0
```

Variable number of arguments

for-of loop enumerates the input arguments

Problem: Incomes and Expenses

- We are given a sequence of **commands**:
  - **Income: {sum}**
  - **Spend: {sum}**
- Write a function to **process all commands** and calculate the final balance (starting with 0 initially)

```
console.log(processExpenses(
  "Income: 50", "Income: 70", "Expense: 30",
  "Income: 100", "Expense: 40", "Income: 50");
// 200
```

Solution: Incomes and Expenses

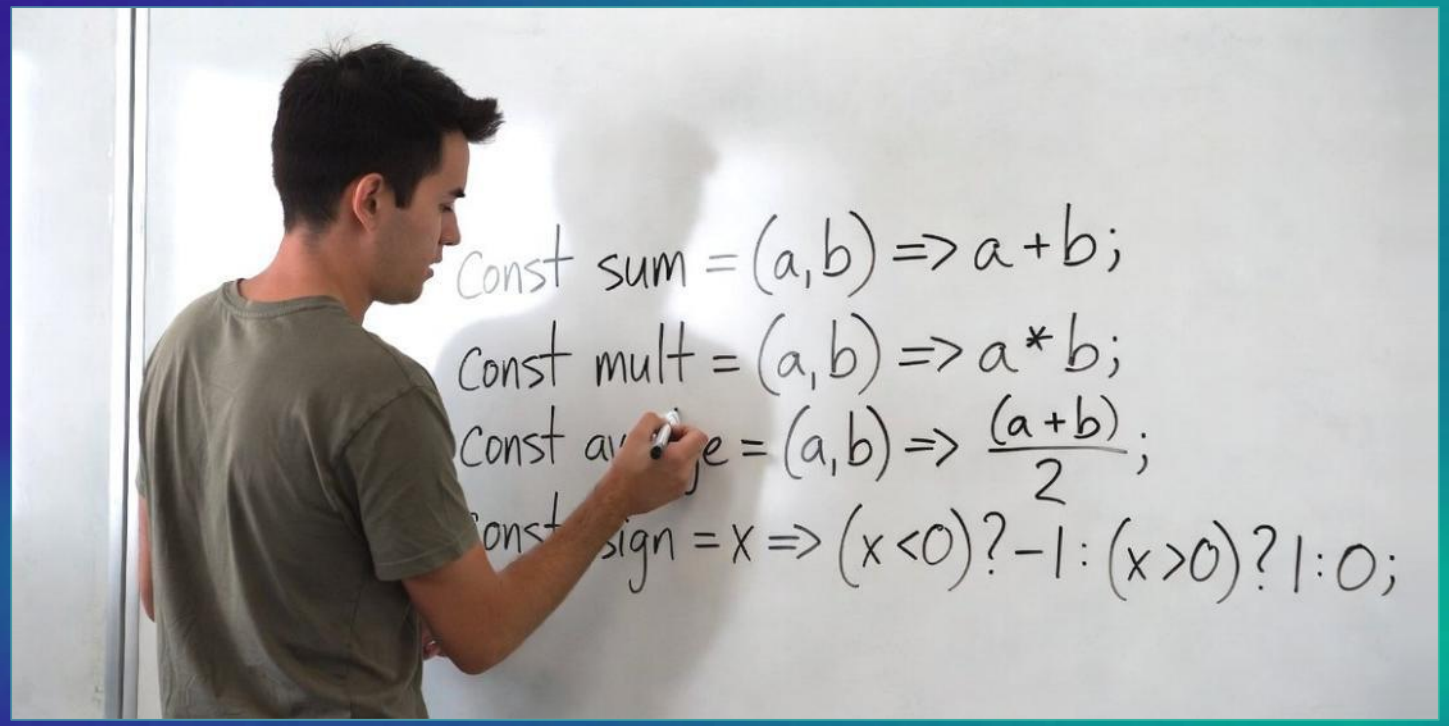
SoftUni AI

```
function processExpenses(...commands) {
  let balance = 0;
  for (let command of commands) {
    const [type, amount] = command.split(": ");
    const value = parseFloat(amount);
    if (type == "Income")
      balance += value;
    else if (type == "Expense")
      balance -= value;
  }
  return balance;
}
```

Judge link: https://alpha.judge.softuni.org/contests/functions-objects-events/5273

Arrow Functions

Function Expressions, Arrow Functions, Higher-Order Functions



Function Expressions

`f = function(…) {…}`

SoftUni AI

- Variables can hold **values** of type "**function**"
- Defined through a **function expression**

```
const add = function(a, b) {
  return a + b;
}

console.log(add); // [Function: add]
console.log(add(2, 3)); // 5

let sum = add, sqrt = Math.sqrt;
console.log(sqrt(sum(8, 1))); // 3
```

Arrow Functions

```
(x) => 2 * x
```

SoftUni AI

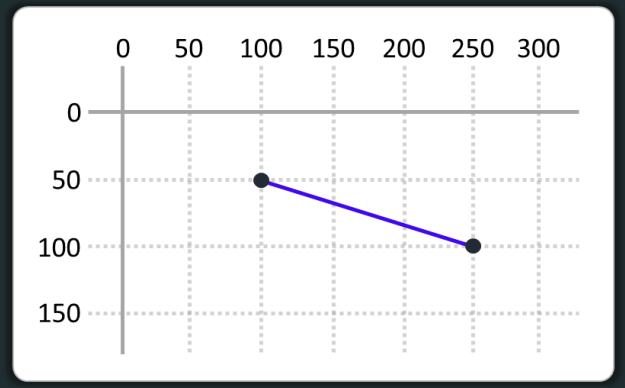
- **Arrow functions** (lambda) use the arrow operator `=>` to provide a **shorter syntax** for function expressions:

```
const sum = (a, b) => a + b;
const mult = (a, b) => a * b;
const average = (a, b) => (a + b) / 2;
const sign = x => (x < 0) ? -1 : (x > 0) ? 1 : 0;

console.log(sum(2, 5), mult(2, 5)); // 7 10
console.log(average(sum(2, 5), mult(2, 5))); // 8.5
console.log(sign(3), sign(0), sign(-4)); // 1 0 -1
```

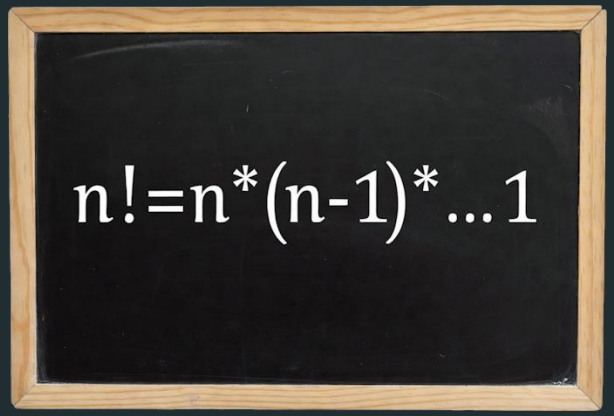
Arrow Functions – More Examples

SoftUni AI

```
const distance = (x1, y1, x2, y2) =>
  Math.sqrt((x2-x1)**2 + (y2-y1)**2);

console.log(distance(100, 50, 250, 100);
// 158.11388300841898
```




```
const factorial = (n) => {
  if (n == 0) return 1;
  return n * factorial(n-1);
}

console.log(factorial(5)); // 120
```



Anonymous Functions

SoftUni AI

- **Anonymous functions** have no name:

```
console.log(function(x) { return x * x });
// [Function (anonymous)] (unnamed function)
```

```
console.log(() => console.log("hello"));
// [Function (anonymous)] (unnamed function)
```

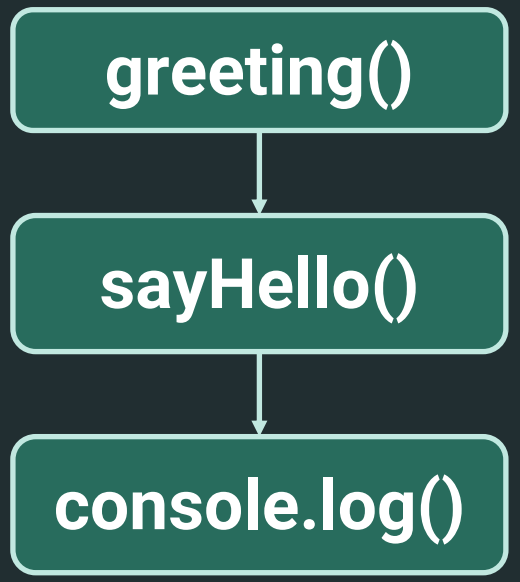
```
let sum = (a, b, c) => a + b + c;
console.log(sum);
// [Function: sum] (has name "sum", not anonymous)
```

Functions as Arguments

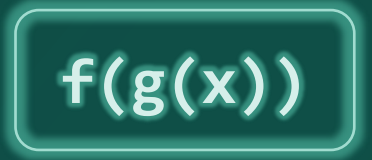
SoftUni AI

- **Functions** can be passed as **arguments** to other functions:

```
const sayHello = () => "Hello, ";
const sayHi = () => "Hi, ";
const sayBye = () => "Bye, ";

function greeting(greetingFunc, name) {
  return greetingFunc() + name;
}

console.log(greeting(sayHello, "JS!")); // Hello, JS!
console.log(greeting(sayHi, "JS!")); // Hi, JS!
console.log(greeting(sayBye, "JS!")); // Bye, JS!
```



Higher-Order Functions



SoftUni AI

- **Higher-order functions** work by invoking other functions, passed as arguments (or return a function as a result)

```
function aggregate(start, end, operation) {
  for (var result = start, i = start+1; i <= end; i++)
    result = operation(result, i);
  return result;
}
```

```
console.log(aggregate(1, 5, (a, b) => a + b)); // 55
console.log(aggregate(1, 5, (a, b) => a * b)); // 120
console.log(aggregate(1, 5, (a, b) => '' + a + b)); // 12345
```

Problem: Special Numbers

SoftUni AI

- Write a function to return all numbers in range [start … end]
  - Divisible to **3**
  - Containing digit **2**
- Use **higher-order function**: iterate over the range in a loop and filter the range with arrow function

```
console.log(specialNumbers(20, 30));
// Nums: 21 24 27
```

```
console.log(specialNumbers(100, 200));
// Nums: 102 120 123 126 129 132 162 192
```

Solution: Special Numbers

SoftUni AI

```
function specialNumbers(start, end) {
  function generateRange(start, end, filter) {
    let result = '';
    for (let num = start; num <= end; num++)
      if (filter(num))
        result += (result ? ' ' : '') + num;
    return result;
  }
  let filterDiv3 = (num) => num % 3 == 0;
  let filterContains2 = (num) => num.toString().includes('2');
  let filters = (num) => filterDiv3(num) && filterContains2(num);
  return "Nums: " + generateRange(start, end, filters);
}
```

Judge link: https://alpha.judge.softuni.org/contests/functions-objects-events/5273

Function Returned by Function

SoftUni AI

- A **function can return another function** as output:

```
function greeting(message) {
  return function(name) {
    return message + name;
  }
}

let sayHi = greeting("Hi, ");
console.log(sayHi); // [Function: anonymous]
console.log(sayHi("Steve")); // Hi, Steve

let sayWelcome = greeting("Welcome, ");
console.log(sayWelcome("Steve")); // Welcome, Steve
```

```
f(x) => function g(x)
```

Closures

SoftUni AI

- **Closure** == function returning a function, with an **internal state**

```
function createCounter(start) {
  let count = start; // Internal state: count
  return function() {
    return count++;
  }
}
let counter1 = createCounter(100);
console.log(counter1(), counter1()); // 100 101
let counter2 = createCounter(1);
console.log(counter2(), counter2()); // 1 2
```

IIFE

SoftUni AI

- **IIFE** == Immediately Invoked Function Expression

```
(function(x) {
  console.log(x * 2);
})(5); // 10
```

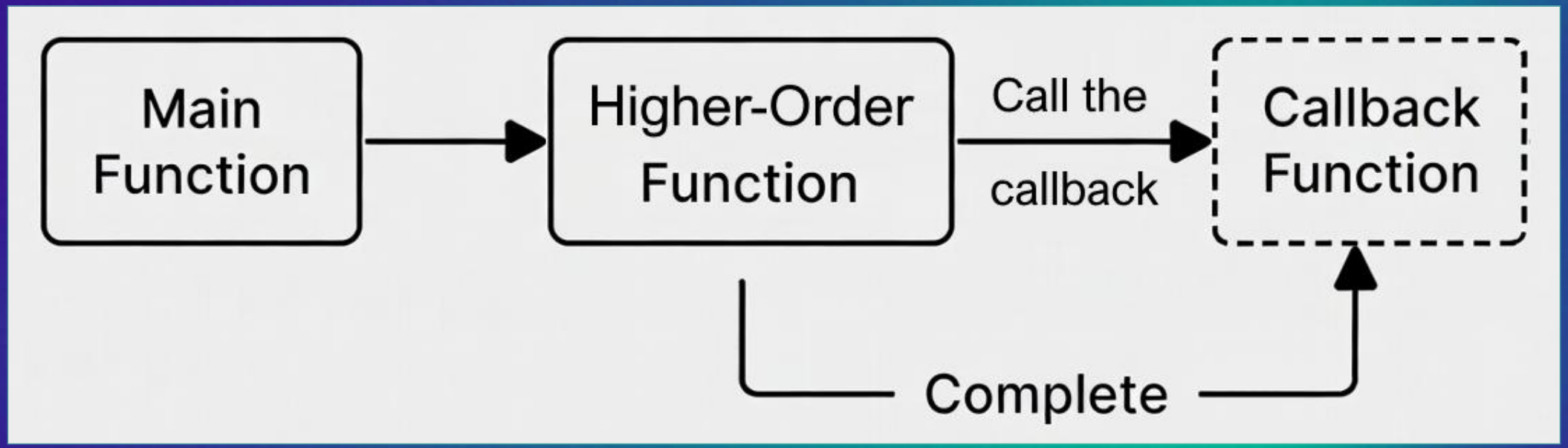
```
((x) => {
  console.log(x * 2);
})(5); // 10
```

- Why to use IFFE? → to **hold data private**

```
(() => {
  let privateData = 42;
  console.log("Inside:", privateData);
})();
console.log("Outside:", typeof privateData); // "undefined"
```

Callback Functions

Functions, Sent as Arguments,
Designed to be "Called Back"



Callbacks

SoftUni AI

- In programming, a **callback function** is a function, given as parameter, indented to be "***called back***"

```
function scanRange(start, end,
    onStart, onNumber, onEnd) {
  onStart(); // Invoke a callback
  for (let i = start; i <= end; i++) {
    onNumber(i); // Invoke a callback
  }
  onEnd(); // Invoke a callback
}
```

Using Callback Functions

SoftUni AI

- Invoking a function, which requires **callback functions**:

```
scanRange(1, 3,
  () => console.log("Starting scan..."),
  (num) => console.log("Found number: " + num),
  () => console.log("Scan complete.")
);
```

```
Starting scan...
Found number: 1
Found number: 2
Found number: 3
Scan complete.
```

Built-In Callbacks in JavaScript

SoftUni AI

- In JavaScript **callbacks** are highly popular, for example:
  - **setTimeout(…)** will invoke a **callback** after time elapsed

```
console.log("Starting a timer…");

function timeoutCallback() {
  console.log("Time's up!");
}

setTimeout(timeoutCallback, 2000);

console.log("Waiting for 2 seconds …");
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  ···
C:\nodejs\node.exe .\timer.js
Starting a timer…
Waiting for 2 seconds …
Time's up!
```

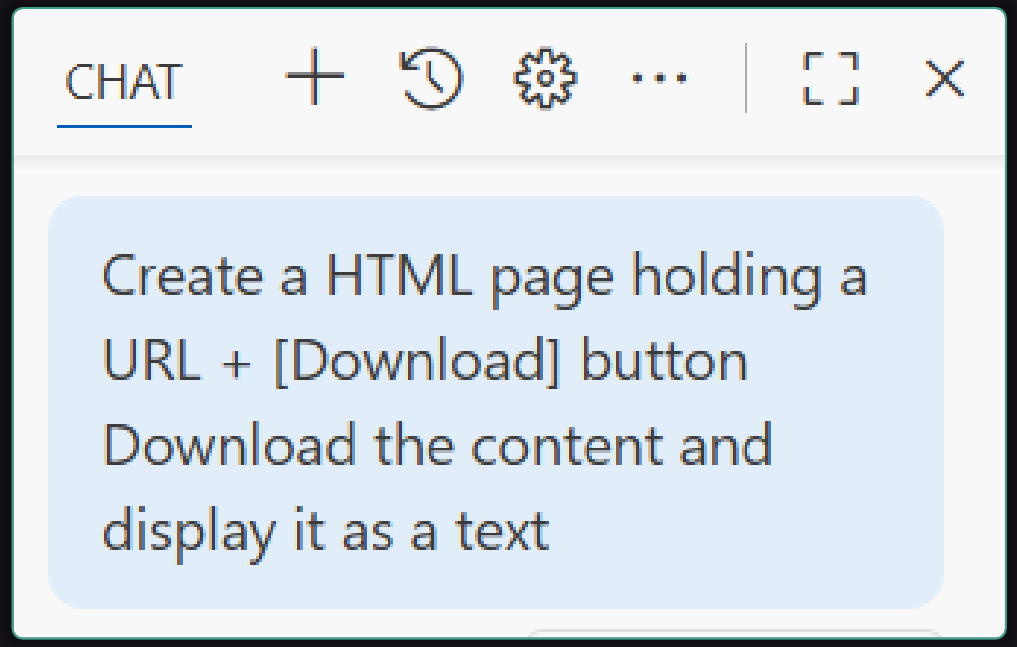
Real-World Callbacks in JS

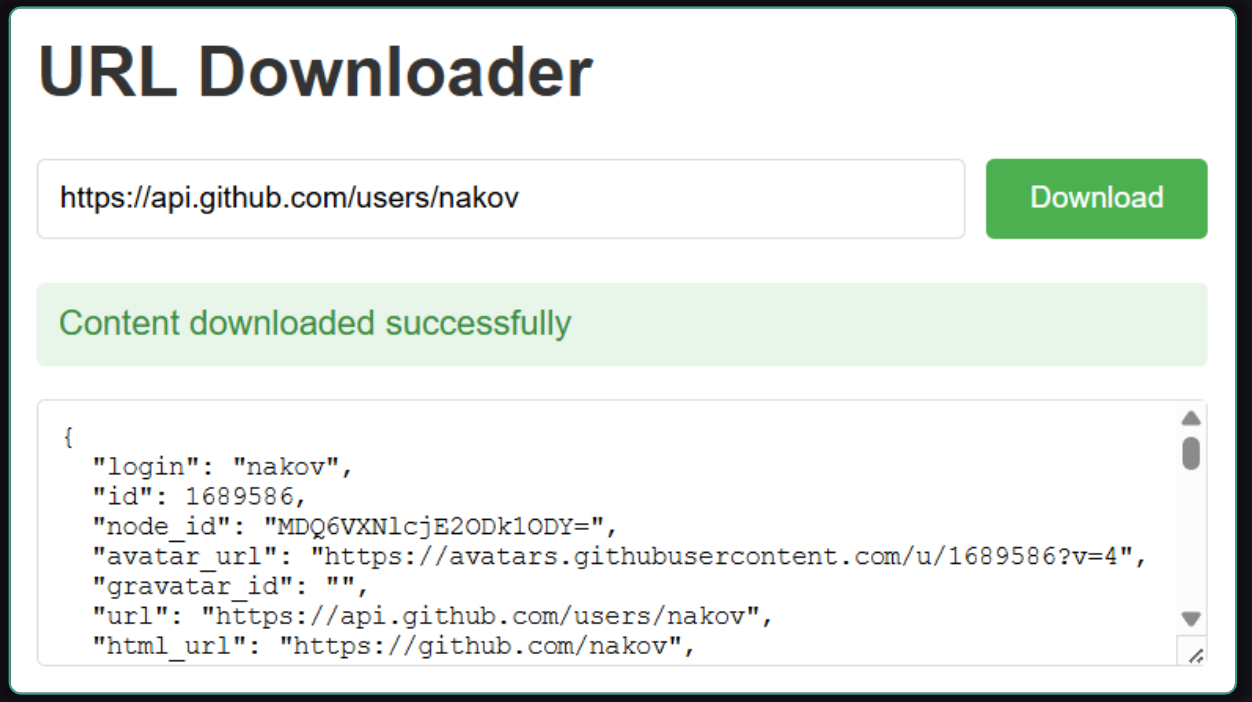
SoftUni AI

```
const https = require('https');
let request = https.get('https://softuni.org');
request.on('response', function(response) {
  let data = '';
  response.on('data', chunk => data += chunk);
  response.on('end', () => console.log(data));
});
request.on('error', function(error) {
  console.log('Network error: ' + error.message);
});
```

Problem: URL Downloader

- Create a **HTML page** holding a **URL** + **[Download]** button
  - Download the content and display it as a text
- **Solution**: prompt **Copilot**

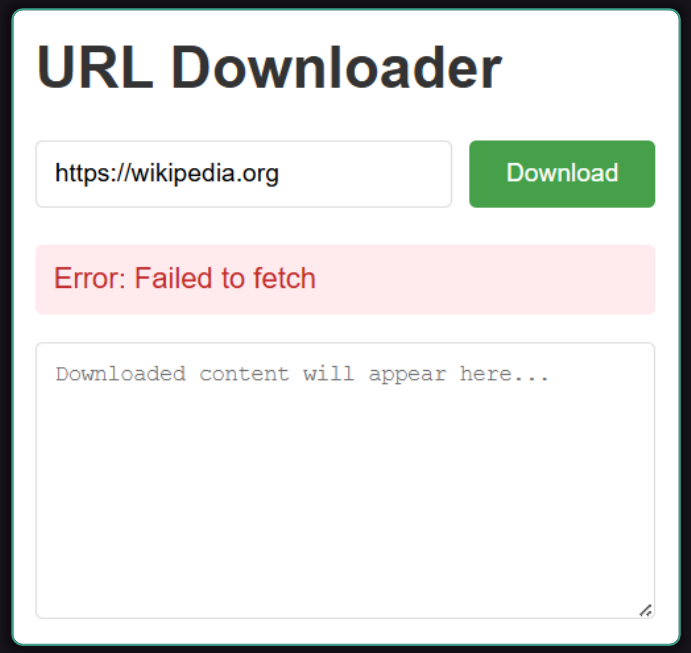




URL Downloader Fails

SoftUni AI

- The "**URL Downloader**" app **fails** to load most Web sites:
  - Loading `**https://wikipedia.org**` → Error: Failed to fetch
- This is a **security limitation** in Web browsers: **CORS policy**



- Use CORS-enabled URL addresses, e.g.
  - https://api.github.com/users/nakov
  - https://catfact.ninja/fact
  - https://api.chucknorris.io/jokes/random
  - https://pokeapi.co/api/v2/pokemon/pikachu

Objects in JavaScript

Keeping a Set of Properties Together

Objects in JavaScript

SoftUni AI

- **Objects** in JavaScript keep a set of properties together

```
let person = {
   name: "Maria",
   age: 24,
   town: "Sofia"
};
```

person	
name	Maria
age	24
town	Sofia

```
person.age++;
console.log(`I am ${person.name}.`);
console.log(`I am ${person.age} years old.`);
```

Working with Objects

SoftUni AI

```
let person = { name: "Maria", age: 24 };
person.isStudent = true; // Add a new property
console.log(person);
// { name: 'Maria', age: 24, isStudent: true }
person.age++; // Change a property
console.log(person['age']); // Access by index → 25
delete person.isStudent; // Remove a property
console.log(person); // { name: 'Maria', age: 25 }
for (let key in person) // Loop through the object
  console.log(key + ": " + person[key]);
```

Nested Objects

SoftUni AI

Nested object: object inside another object

```
const user = {
  name: "George",
  address: { town: "Plovdiv", country: "BG" },
};
console.log(user);

user.username = "gogo98";
let job = { title: "Developer", company: "SoftUni" }
user.job = job;
user.job.title = "CTO";
console.log(user); // { name : ..., address: ..., username: ..., job: ... }
```

Objects as Function Parameters

SoftUni AI

```
function printUser({name, address: {town, country}}) {
  console.log(`Name: ${name}`);
  console.log(`City: ${town}`);
  console.log(`Country: ${country}`);
}

const user = {
  name: "Nina",
  address: { town: "Plovdiv", country: "BG" }
};
printUser(user);
```

This syntax is called "object destructuring"

Problem: Calculate Stats

SoftUni AI

- Write a function to take **several numbers** as input and return an **object**, holding **statistics** about the numbers:
  - **Minimal** number, **maximal** number, **average** of the numbers, **sum** of the numbers

Input:	`calcStats(5, 10, 15, 20, 25)`
Output:	`{ min: 5, max: 25, average: 15, sum: 75 }`

Input:	`calcStats(2.50, -7.25, 17500.00, 0.25, 400, -0.5)`
Output:	`{ min: -7.25, max: 17500, average: 2982.5, sum: 17895 }`

Solution: Calculate Stats

- Functions can **return objects** as output:

```
function calcStats(...numbers) {
  let [min, max, sum] =
    [Number.POSITIVE_INFINITY, Number.NEGATIVE_INFINITY, 0];
  for (num of numbers) {
    if (num < min) min = num;
    if (num > max) max = num;
    sum += num;
  }
  return { min, max, average: sum / numbers.length, sum };
}
```

Group declaration + assignment

Return an object

Judge link: https://alpha.judge.softuni.org/contests/functions-objects-events/5273

Problem: Largest Rectangle

- Write a function to take **several rectangles**, given as **objects {width, height}** and print the **largest** of them

```
printLargestRectangle(
 {width:30, height:20},
 {width:5, height:120},
 {width:15, height:40},
 {width:25, height:25},
 {width:35, height:15}
);
```

```
Largest rectangle: 25 x 25 -> area: 625
```

Solution: Largest Rectangle

SoftUni AI

```
function printLargestRectangle(...rectangles) {
  let largestArea = 0, largestRect = null;
  for (let rect of rectangles) {
    const area = rect.width * rect.height;
    if (area > largestArea)
      [largestArea, largestRect] = [area, rect];
  }
  if (largestRect)
    console.log(`Largest rectangle: ${largestRect.width} x
${largestRect.height} -> area: ${largestArea}`);
}
```

Judge link: https://alpha.judge.softuni.org/contests/functions-objects-events/5273

Methods in Objects

Implementing Logic with the Object State

object	`const rectangle = {`
state	`x: 150, y: 40,` `width: 20, height: 15,`

methods	`move: function(dx, dy) { … },`
	`calcArea: function() { … },`
	`toString: function() { … }`
	`}`

Methods: Functions inside Objects

SoftUni AI

```
const rectangle = {
  x: 150, y: 40,
  width: 20, height: 15,
  move: function(dx, dy) {
    this.x += dx;
    this.y += dy;
  },
  calcArea() { return this.width * this.height; },
  toString() { return `Rect(${this.x}, ${this.y})`; }
}
```

Method move(dx, dy)

`this` means the object itself

Method toString()

Calling Methods of an Object

SoftUni AI

```
console.log("" + rectangle); // Invokes toString()
// Rect(150, 40)

rectangle.move(50, -10);
console.log(rectangle.toString()); // Rect(200, 30)

console.log("Area:", rectangle.calcArea()); // Area: 300

for (let i=10; i<=100; i+=10) {
  rectangle.move(i, 0);
  console.log("Rectangle moved to: " + rectangle);
}
```

JSON

SoftUni AI

- **JSON** is a text-format for storing JavaScript objects

```
const user = {
  name: "George",
  address: { town: "Plovdiv", country: "BG" },
};

const userJSON = JSON.stringify(user);
console.log(userJSON); // typeof(userJSON) → 'string'

const colorJSON = '{"red":75,"green":89,"blue":43}';
let c = JSON.parse(colorJSON);
console.log(`RGB(${c.red}, ${c.green}, ${c.blue})`);
```

Problem: Moving Points

- Write a JS function, to take as input a **JSON point {x, y}** and several **JSON moves {dx, dy}** and applies the moves

```
movePoints(
   '{"x":50, "y":100}',
   '{"dx":20, "dy":30}',
   '{"dx":-10, "dy":20.5}'
)
```

Note: these arguments are JSON strings, not objects!

```
Initial point: (50, 100)
Moved point: (70, 130)
Moved point: (60, 150.5)
```

Solution: Moving Points

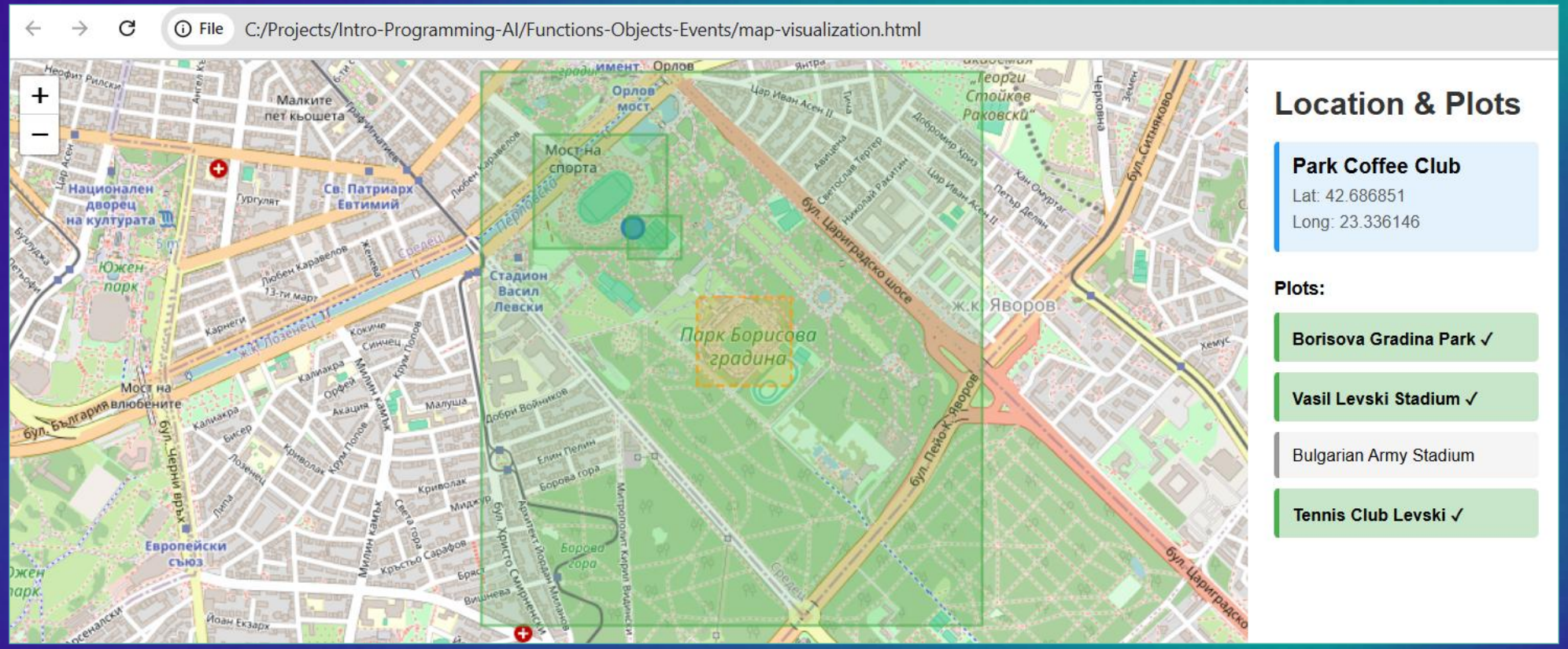
SoftUni AI

```
function movePoints(pointJSON, ...movesJSON) {
  let point = JSON.parse(pointJSON);
  console.log(`Initial point: (${point.x}, ${point.y})`);
  point.move = function(dx, dy) { this.x += dx; this.y += dy; };
  point.toString = function() { return `(${this.x}, ${this.y})`; };
  for (const moveJSON of movesJSON) {
    let move = JSON.parse(moveJSON);
    point.move(move.dx, move.dy);
    console.log("Moved point: " + point);
  }
}
```

Judge link: https://alpha.judge.softuni.org/contests/functions-objects-events/5273

Plots on an Interactive Map

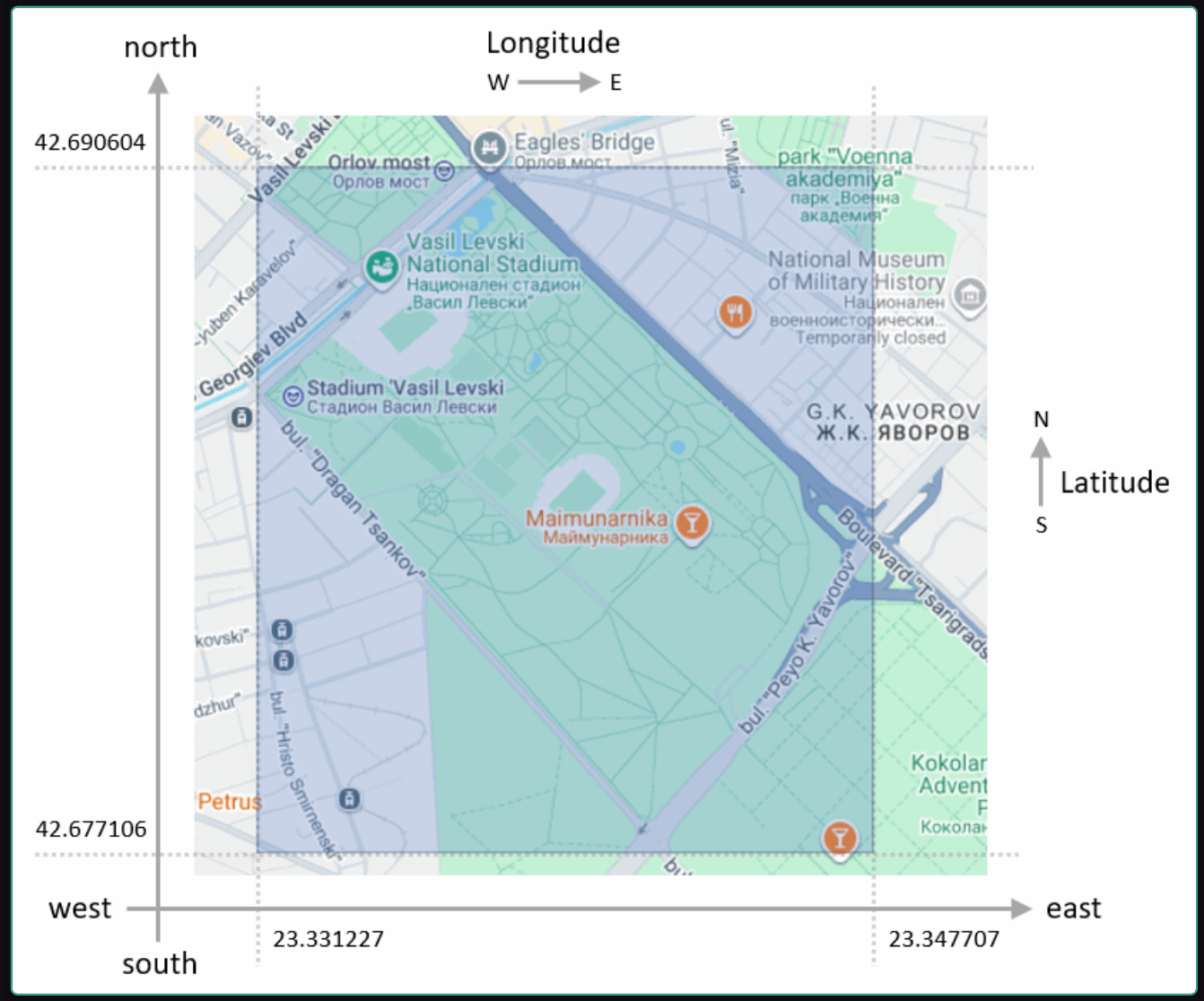
Real-World Problem



Intro to Geographic Coordinates

SoftUni AI

- On the **geographical maps** points have unique **geographic coordinates** (GPS coordinates):
  - **Latitude**: south → north (-90° ... +90°)
  - **Longitude**: west → east (-180° ... +180°)
- Rectangles are defined as:
  - **{SW, NE}** coordinates



Problem: Plots of Land

SoftUni AI

- We are given a **GPS location**:
  - JS object: **{location, lat, long}**
  - Example: **{name:"Park Coffee Club", lat:42.686851, long:23.336146}**
- We are also given several **plots of land** (rectangular areas, for simplicity), with their **GPS coordinates**:
  - JS object: **{name, S, W, N, E}**
  - Example: **{name:"Vasil Levski Stadium", S: 42.686344, W: 23.332922, N: 42.689106, E: 23.337278}**
- Write a function to print all **plots that overlap** with the location

Plots of Land – Example

SoftUni AI

```
function printCrossingPlots(location, ...plots) { … }
```

```
printCrossingPlots(
 {location:"Park Coffee Club", lat: 42.686851, long: 23.336146},
 {name:"Borisova Gradina Park", S: 42.677265, W: 23.331184, N: 42.690606, E: 23.347621},
 {name:"Vasil Levski Stadium", S: 42.686344, W: 23.332922, N: 42.689106, E: 23.337278},
 {name:"Bulgarian Army Stadium", S: 42.683029, W: 23.338244, N: 42.685191, E: 23.341355},
 {name:"Tennis Club Levski", S: 42.686102, W: 23.336007, N: 42.687132, E: 23.337734}
);
```

```
Crossing plots of Park Coffee Club:
 - Borisova Gradina Park
 - Vasil Levski Stadium
 - Tennis Club Levski
```

Solution: Plots of Land

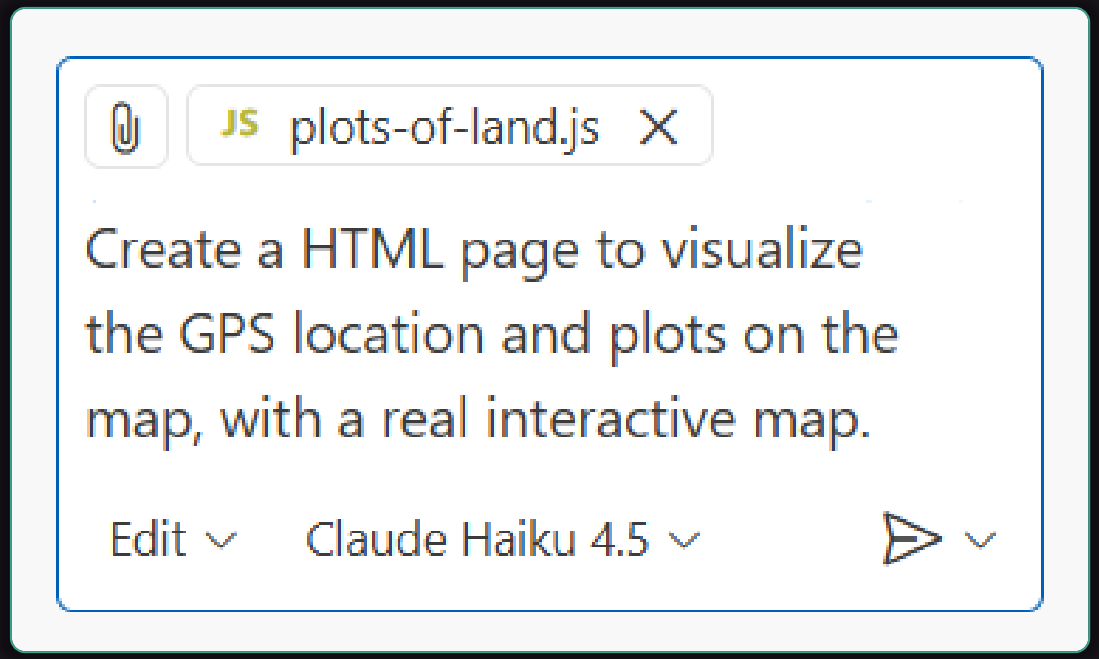
SoftUni AI

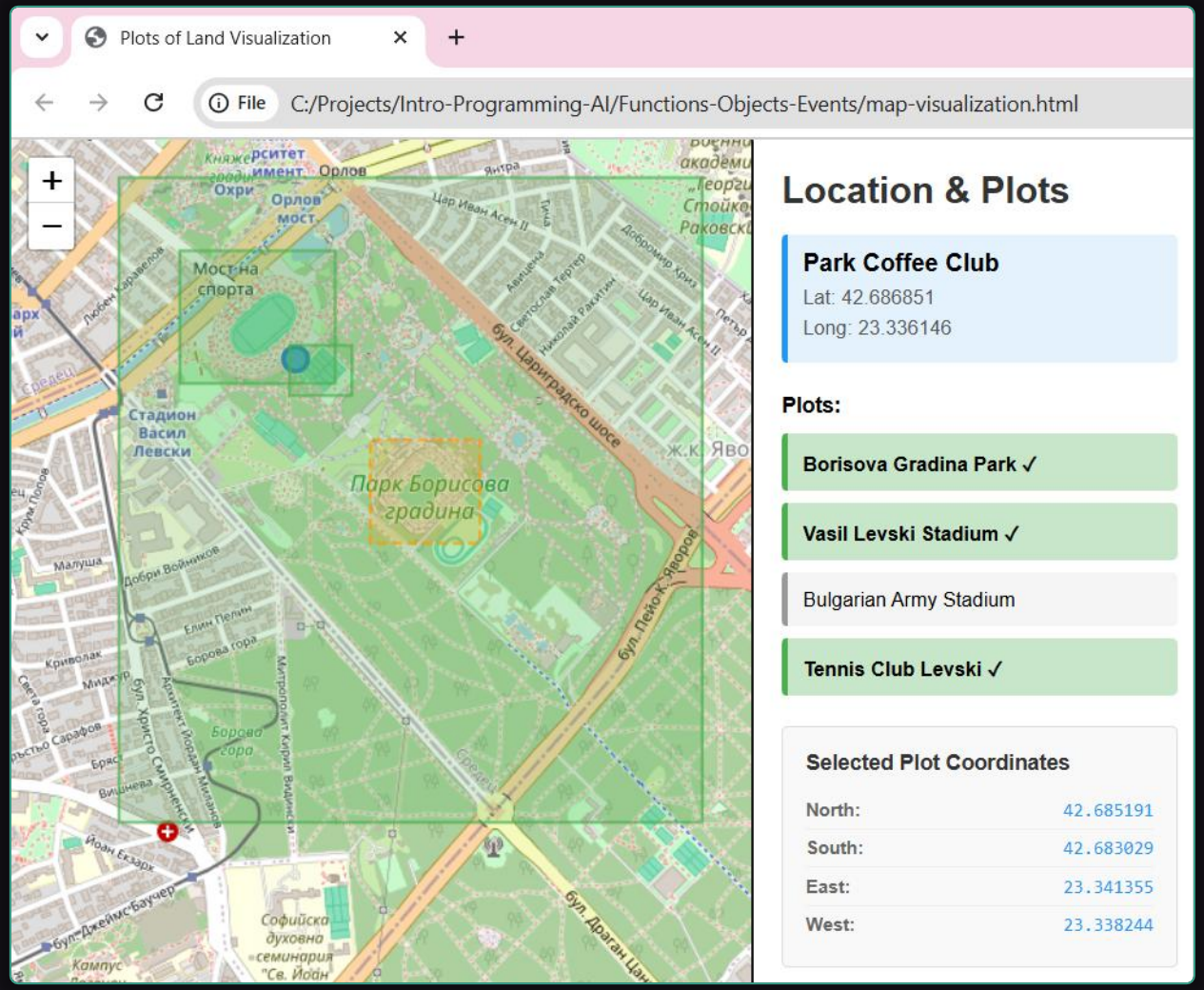
```
function printCrossingPlots({location, lat, long}, ...plots) {
  console.log("Crossing plots of " + location + ":");
  for (let plot of plots) {
    if (lat >= plot.S && lat <= plot.N &&
        long >= plot.W && long <= plot.E) {
      console.log(` - ${plot.name}`);
    }
  }
}
```

Judge link: https://alpha.judge.softuni.org/contests/functions-objects-events/5273

Plot of Land – Visualization

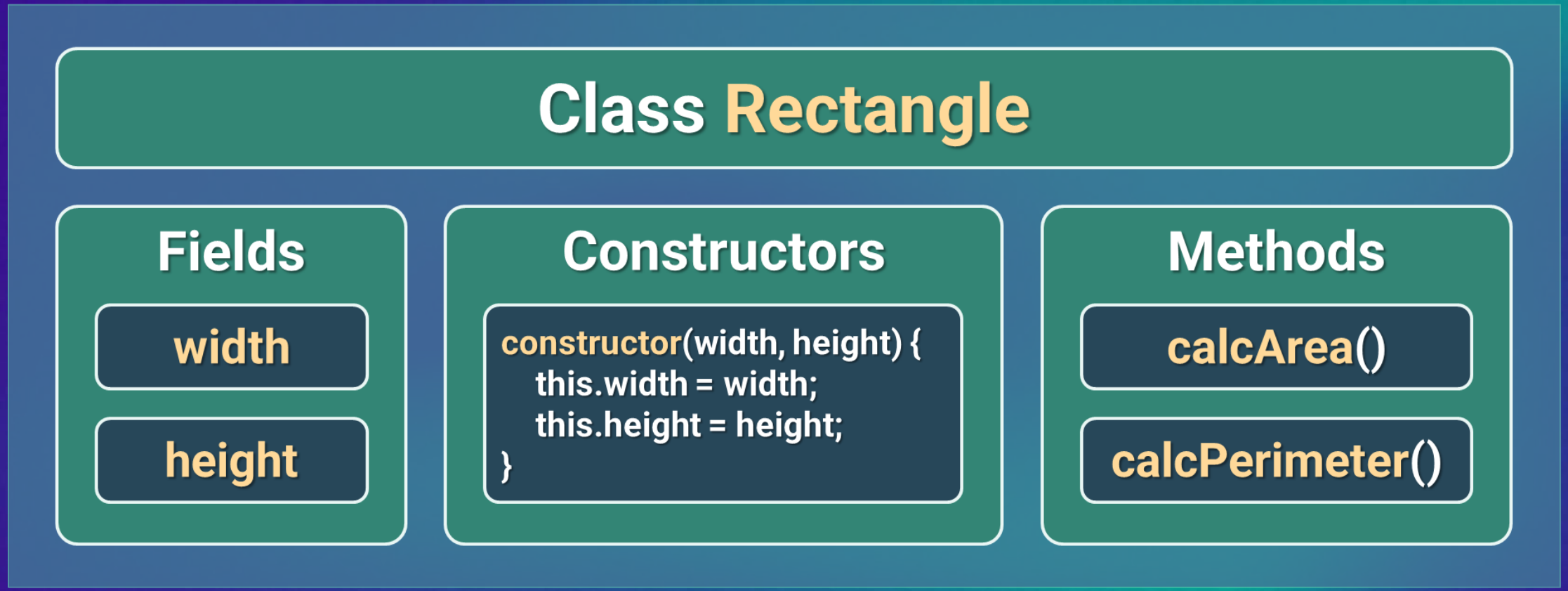
- **Visualize as HTML page**:
  - View the location and plots on an interactive map





Intro to Classes

Classes, Properties, Constructors, Methods



Class Structure

- **Classes** define blueprints for **objects** of the same type
  - Classes hold **properties** (data fields – key-value pairs), **constructors** (field initializers) and **methods** (actions)

Class Rectangle

Properties

- width
- height

Constructors

```
constructor(width, height) {
    this.width = width;
    this.height = height;
}
```

Methods

- calcArea()
- calcPerimeter()

Classes – Example

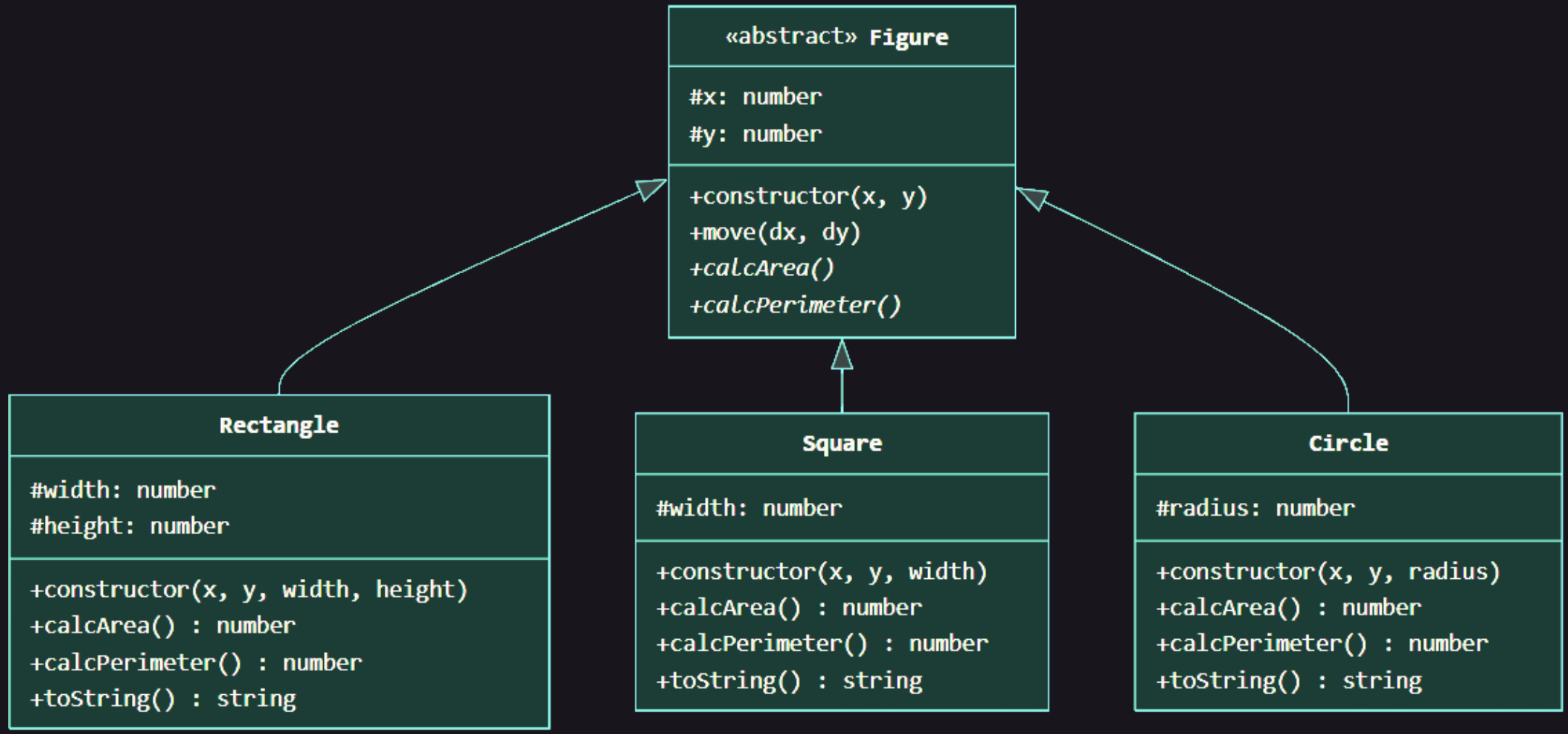
class Rectangle	
state	width, height
constructor	width, height
methods	calcArea() calcPerimeter()

```
class Rectangle {
  constructor(width = 0, height = 0) {
    this.width = width;
    this.height = height;
  }
  calcArea() { return this.width * this.height; }
  calcPerimeter() { return 2 * (this.width + this.height); }
}
```

```
let rect1 = new Rectangle(30, 20);
console.log(rect1, rect1.calcArea(), rect1.calcPerimeter());
let rect2 = new Rectangle(40, 15);
console.log(rect2, rect2.calcArea(), rect2.calcPerimeter());
```

Class Hierarchies

- We can build **hierarchies of classes**, by using **inheritance**



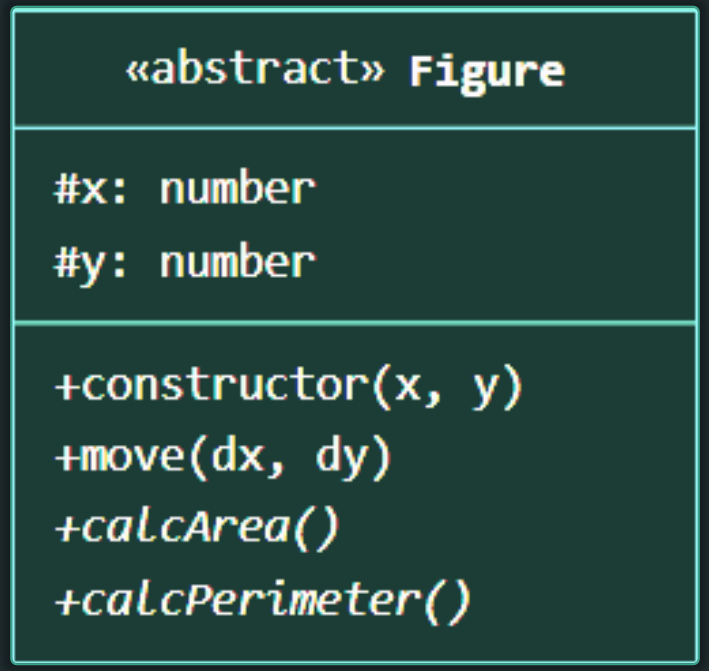
Abstract Base Class Figure

SoftUni AI

```
class Figure {
  constructor(x = 0, y = 0) {
    this.x = x;
    this.y = y;
  }
  move(dx, dy) {
    this.x += dx;
    this.y += dy;
  }
  calcArea() { throw new Error("Can't call abstract method"); }
  calcPerimeter() { throw new Error("Can't call abstract method"); }
}
```

Shared method:
inherited by subclasses

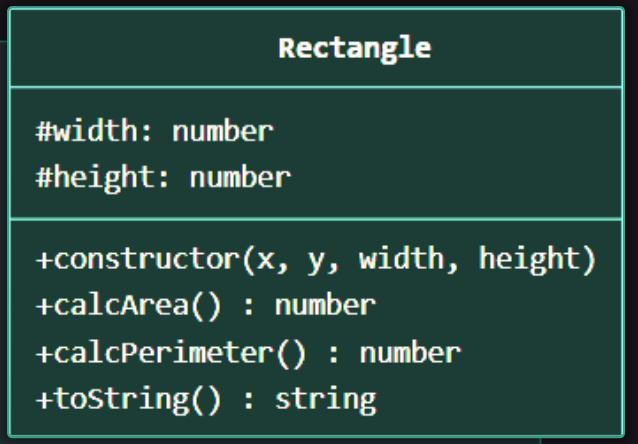
Abstract methods:
should be implemented in subclasses



Class Rectangle

SoftUni AI

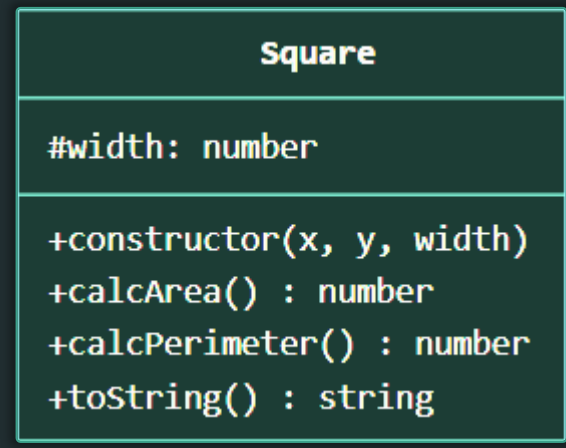
```
class Rectangle extends Figure {
  constructor(x = 0, y = 0, width = 0, height = 0) {
    super(x, y);
    this.width = width;
    this.height = height;
  }
  calcArea() { return this.width * this.height; }
  calcPerimeter() { return 2 * (this.width + this.height); }
  toString() { return `Rectangle(${this.x}, ${this.y}, size
    = ${this.width} x ${this.height})`; }
}
```



Class Square

SoftUni AI

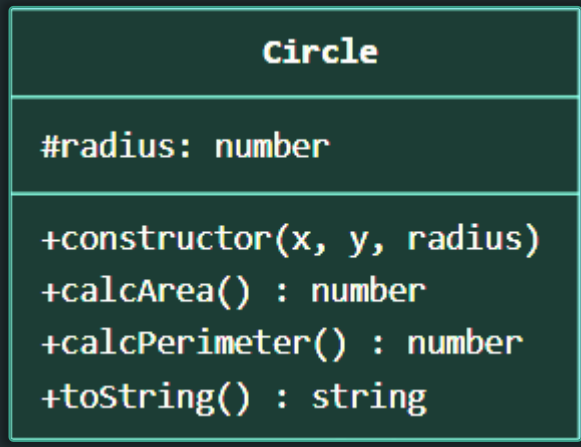
```
class Square extends Figure {
  constructor(x = 0, y = 0, width = 0) {
    super(x, y);
    this.width = width;
  }
  calcArea() { return this.width * this.width; }
  calcPerimeter() { return 4 * this.width; }
  toString() { return `Square(${this.x}, ${this.y},
    width = ${this.width})`; }
}
```



Class Circle

SoftUni AI

```
class Circle extends Figure {
  constructor(x = 0, y = 0, radius = 0) {
    super(x, y);
    this.radius = radius;
  }
  calcArea() { return Math.PI * this.radius ** 2; }
  calcPerimeter() { return 2 * Math.PI * this.radius; }
  toString() {return `Circle (${this.x}, ${this.y},
    radius = ${this.radius})`; }
}
```



Using the Defined Classes

SoftUni AI

```
let r = new Rectangle(10, -5, 10, 5);
r.move(10, -10);
console.log(r);
console.log(r.toString(), r.calcArea(), r.calcPerimeter());

let s = new Square(150, 30, 10);
s.move(-20, 30);
console.log(s.toString(), s.calcArea(), s.calcPerimeter());

let c = new Circle(120, 80, 3);
c.move(0, -5);
console.log(c.toString(), c.calcArea(), c.calcPerimeter());
```

Problem: Largest Figures

- Using the classes **Figure**, **Rectangle**, **Square** and **Circle**, write a JS function
  - Take several **figures** (passed as parameters)
  - Print the figure with the **largest area**
  - Print the figure with the **largest perimeter**

```
printLargestFigures(new Rectangle(10, 15, 300, 120),
  new Square(-20, -50, 200), new Circle(-50, 20, 130));
```

```
Largest area: 53092.92 -> Circle (-50, 20, radius = 130)
Largest perimeter: 840.00 -> Rectangle(10, 15, size = 300 x 120)
```

Solution: Largest Figures

SoftUni AI

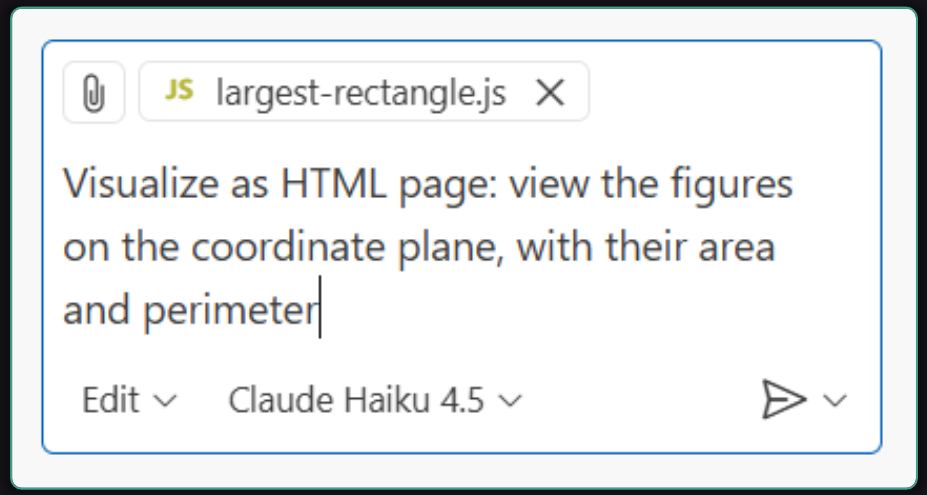
```
function printLargestFigures(...figures) {
  let largestArea = null, largestPerimeter = null;
  for (let f of figures) {
    if (largestArea == null || f.calcArea() > largestArea.calcArea())
      largestArea = f;
    if (largestPerimeter == null ||
        f.calcPerimeter() > largestPerimeter.calcPerimeter())
      largestPerimeter = f;
  }
  console.log(`Largest area: ${largestArea.calcArea().toFixed(2)}
    -> ${largestArea.toString()}`);
  console.log(`Largest perimeter: ${largestPerimeter.calcPerimeter()
    .toFixed(2)} -> ${largestPerimeter.toString()}`);
}
```

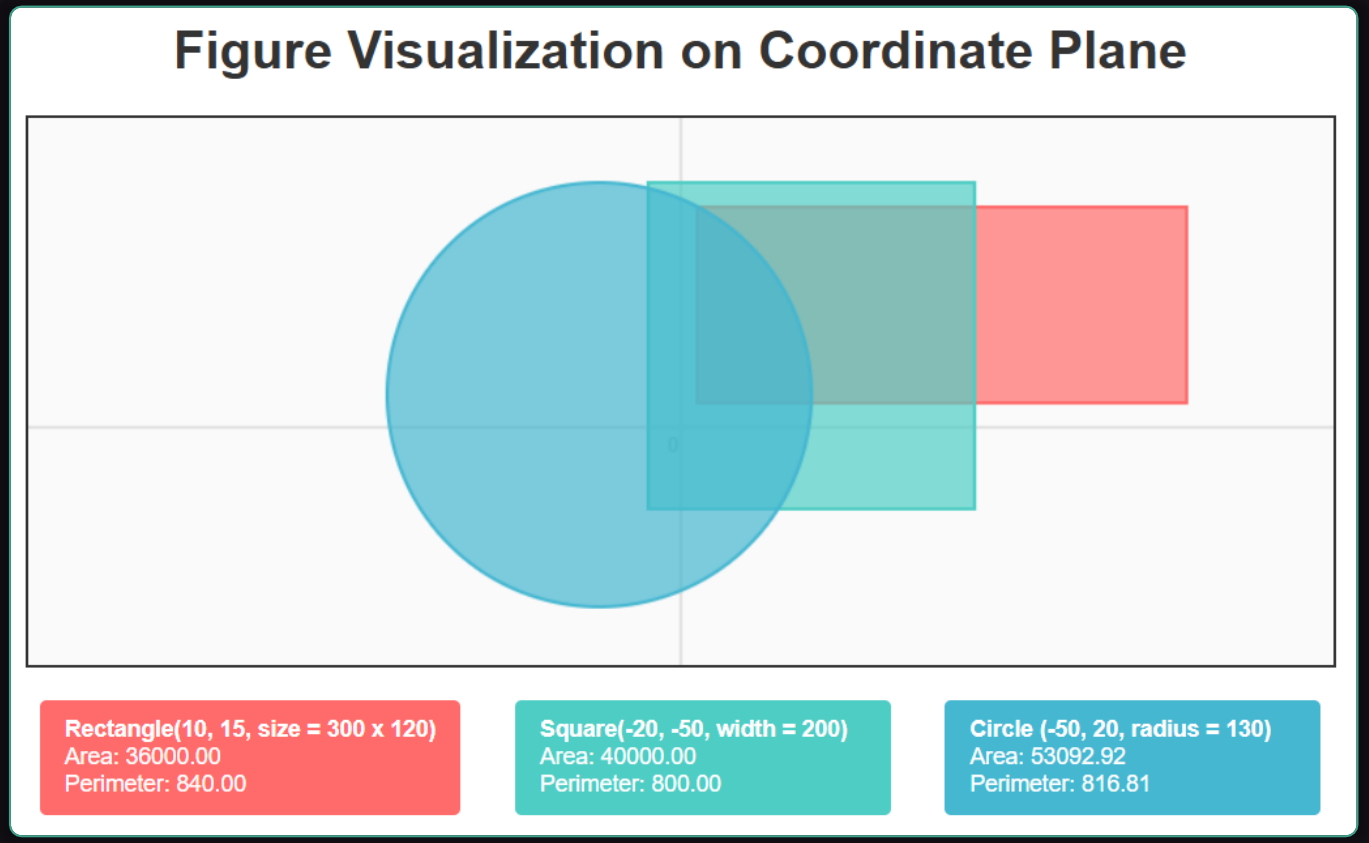
Note: this code cannot be tested in the Judge system

Largest Figures – Visualization

SoftUni AI

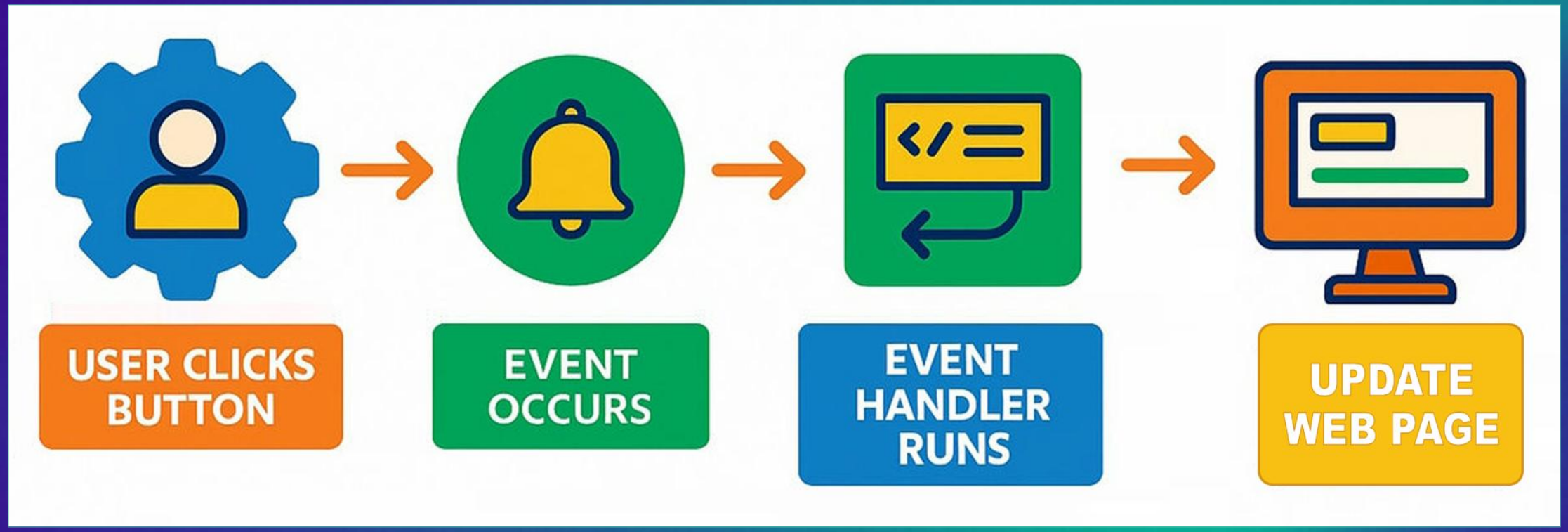
- **Visualize as HTML page**: view the figures on the coordinate plane, with their area and perimeter
- Use a prompt in GitHub Copilot:





Events and Event Handling

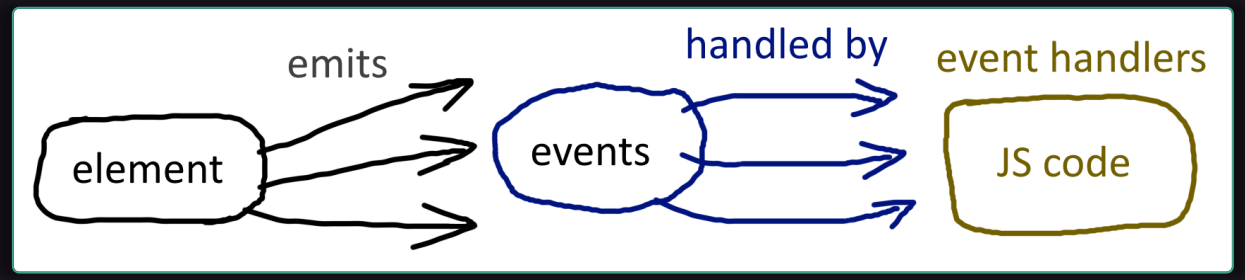
Events in HTML and Event Handlers in JS



Events and Event Handlers

SoftUni AI

- **HTML** pages hold **elements**: document, form fields, buttons, images, texts, …

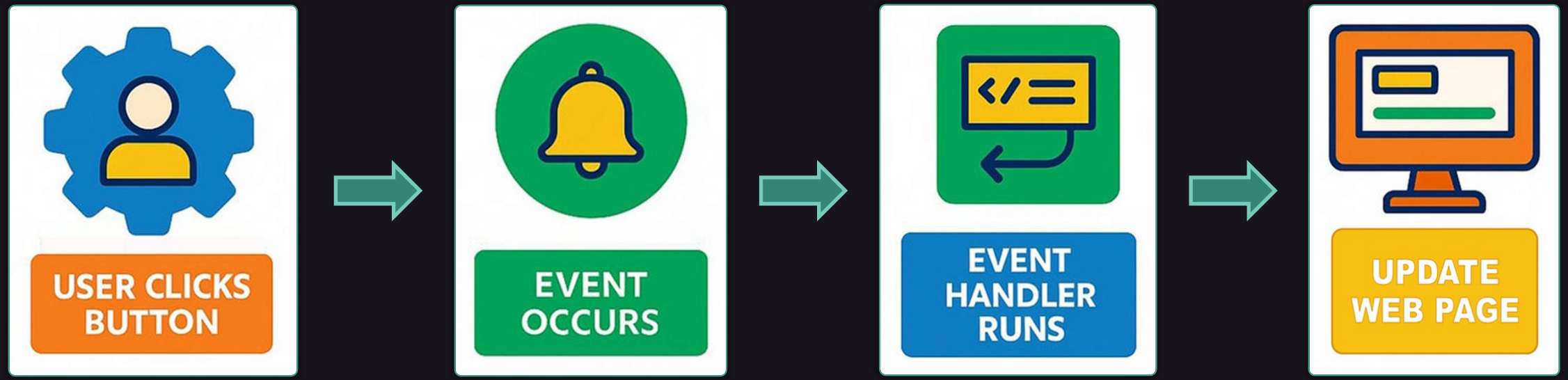


- An **event** is some action that happens in the browser, e.g.
  - The **page is loaded**, a **button is clicked**, a form **field is changed**, a **form is submitted**, the **mouse is moved**, etc.
- Elements **emit events** when certain actions happen
- Events are handled by **events handlers** – JS functions (code) that process the occurred event

Event Handling Lifecycle

SoftUni AI

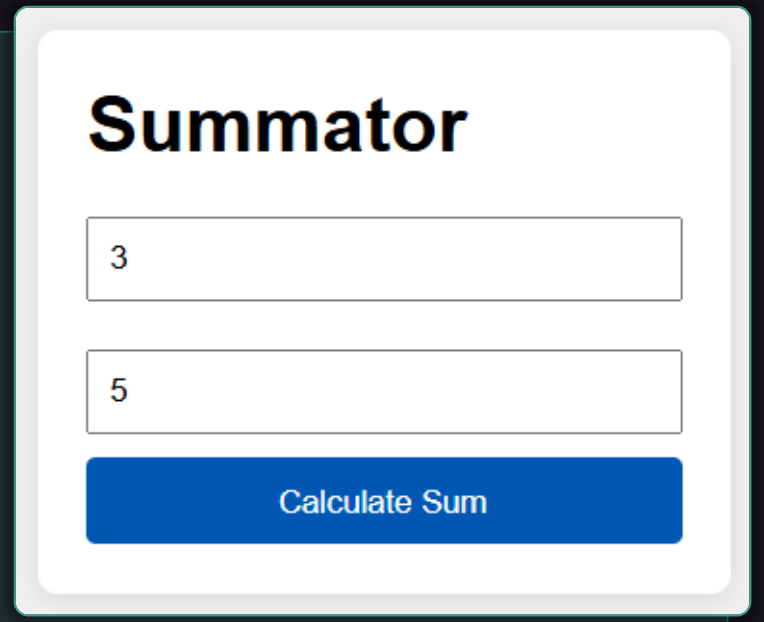
- Event in the Web browser – example:



Event Handling in HTML – Example

SoftUni AI

```
<html>
<head><title>Summator</title></head>
<body>
  <h1>Summator</h1>
  <input type="number" id="num1">
  <input type="number" id="num2">
  <button onclick="sum()">Calculate Sum</button>
  <div id="result"></div>
</body>
</html>
```

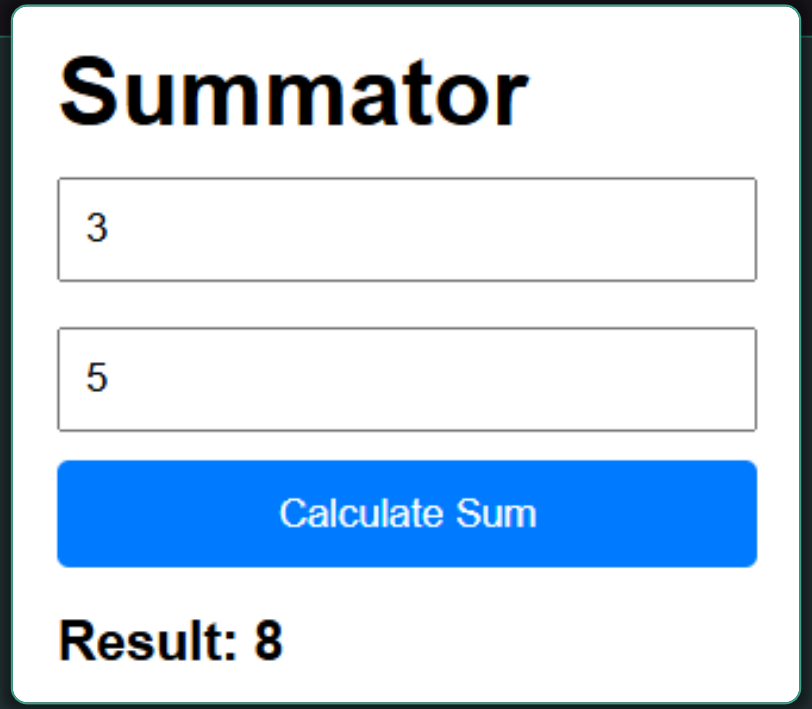


Event "onclick" handled by JS function "sum()"

Event Handling in HTML – Example

SoftUni AI

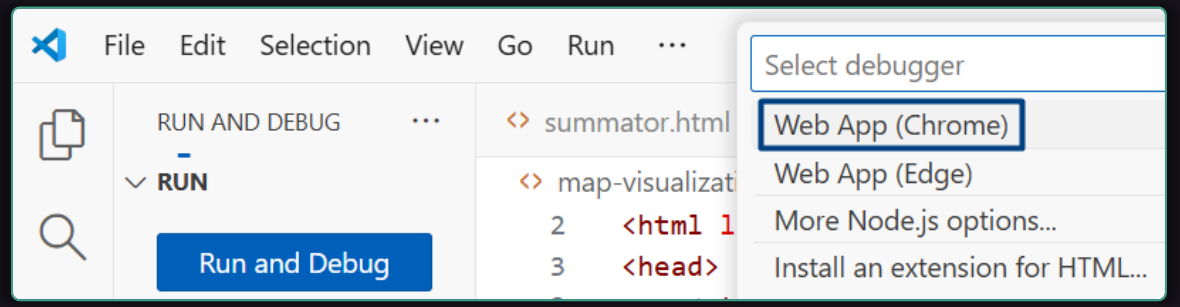
```
<script>
  function sum() {
    let num1 = parseFloat(document.
      getElementById('num1').value);
    let num2 = parseFloat(document.
      getElementById('num2').value);
    let total = num1 + num2;
    document.getElementById('result').
      textContent = `Result: ${total}`;
  }
</script>
```

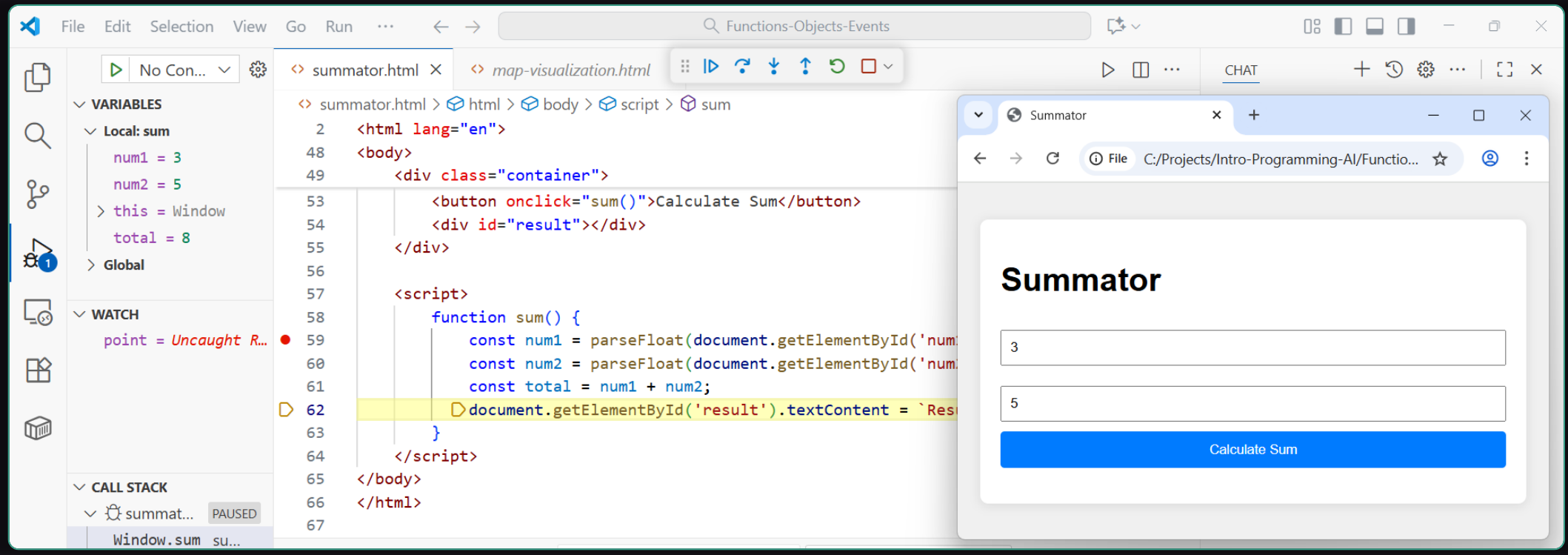


Debugging HTML + JS

SoftUni AI

- We can **debug HTML pages** with JavaScript code from the VS Code Debugger:





Lesson Summary

SoftUni AI

- **Functions** in JavaScript hold blocks of code, can take parameters, and return result
- **Arrow functions** give short syntax: **(a, b) => a + b**
- **Callback** == function passed to another function
- **Objects** hold key-value pairs: **{ name: "Mia", age: 25 }**
- **Classes** define template for objects: **new Circle(25)**
  - Constructors, properties, methods, this
- **Events** are actions, accruing when something happens, handled by **event handlers**: JS callback functions

Postbank – Exclusive Partner for SoftUni AI

- One of the leading **banking institutions** in Bulgaria
- Member of the Eurobank Group with € 99.6 billion of assets
- Innovative trendsetter in the digitalization and transformation
- Certified Top Employer 2024 by the international Top Employers Institute
- AI integration for business, learning and development
- AI Assistants for talents: Story Builder, CV Assistant, Interview Trainer
- Proven people care and wellbeing initiatives
- Benefits and unlimited access to professional, personal and leadership trainings and programs
- www.postbank.bg / careers.postbank.bg

Diamond Partners of Software University

Software University

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025

FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

encorp.ai

createX

DRAFT KINGS

THE CROWN IS YOURS

INDEAVR

Serving the high achievers

VIVACOM

Diamond Partners of SoftUni Digital

SoftUni Digital

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025

FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

encorp.ai

createX

DRAFT KINGS

THE CROWN IS YOURS

INDEAVR

Serving the high achievers

VIVACOM

Diamond Partners of SoftUni Digital

SoftUni Digital

HUMAN

IMPULSE MEDIA®

1 FOR FIT

NETPEAK
DIGITAL GROWTH PARTNER

ABC DESIGN & COMMUNICATION

Zahara
dig.it.all

ETIENYANEV
Break Your *Limits*. Live Your *Brand*.

Diamond Partners of SoftUni Creative

SoftUni Creative

Coca-Cola HBC Bulgaria

Postbank

top EMPLOYER България Bulgaria 2025

FOR A BETTER WORLD OF WORK

SUPER HOSTING .BG

encorp.ai

createX

DRAFT KINGS

THE CROWN IS YOURS

INDEAVR

Serving the high achievers

VIVACOM

Organization Partners of SoftUni Creative

SoftUni Creative

dequitas.

Фигмаистор

Studio π

Think Big • Design Smart

дизайнът на нещата

REDV PRINT STUDIO

THE BUCKS TOWN'S WORK